STM32CubeIDE 中基于 STM32L5 系列的项目入门

引言

本应用笔记描述如何在意法半导体的 STM32CubeIDE 集成开发环境中开始基于 STM32L5 Series 微控制器的项目。

1 概述

STM32CubeIDE 支持基于 Arm® Cortex®处理器的 STM32 32 位产品。

提示 *Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.*

1.1 先决条件

以下工具是理解本文档中的教程并开发基于 STM32L5 Series（Arm® TrustZone®已启用）的应用程序的先决条件：

- STM32CubeIDE 1.2.0
- STM32CubeProgrammer (STM32CubeProg) 2.3.0: 选项字节配置
- STM32Cube_FW_L5_V1.0.0：STM32CubeL5 固件与示例项目、以及 HAL 和 CMSIS 驱动程序

建议用户保持关注 www.st.com/en/microcontrollers-microprocessors/stm32l5-series 上的 STM32L5 Series 文档演化。

1.2 本文档中的用例

在 STM32CubeIDE 上下文中，用户有许多不同的方法来探索和着手开发基于 STM32L5 Series 的项目：

- 从 STM32CubeL5 MCU 开发包导入一个 STM32CubeIDE 项目，以通过工作示例来学习
- 使用 STM32CubeMX 工具创建 STM32CubeMX 项目
- 在 STM32CubeIDE 中创建一个空项目，编写自己的代码
- 在 STM32CubeIDE 中创建一个空项目，从 STM32CubeL5 MCU 开发包中复制示例项目代码到创建的项目中

建议采用以下方法熟悉和开始基于 STM32L5 Series 的项目开发：

1. 导入一个 TrustZone®示例项目，该项目是 STM32CubeL5 MCU 开发包的组成部分。这是了解 STM32L5 MCU 启动部分代码 CMSIS 和 HAL 驱动程序的最快方法。
2. 创建一个空项目，并从 STM32CubeL5 MCU 开发包复制代码。在空项目中，用户完全控制源代码和配置文件，STM32CubeMX 则无法触及。这为用户提供了更高的灵活性，但需要稍微陡一些的学习曲线。
3. 创建一个 STM32CubeMX 项目，使用图形界面配置硬件并生成相应的 HAL 驱动程序。这可以作为量产项目或学习项目，以便进行深入探索和学习。

一些模板项目以 STM32CubeIDE 项目的格式提供；这些项目是已启用或未启用 TrustZone®的模板项目。例如：

- 使用 `TZEN = 1`：

  `STM32Cube_FW_L5_V1.0.0\STM32Cube_FW_L5_V1.0.0\Projects\STM32L552E-EV\Templates\TrustZoneEnabled\`
- 使用 `TZEN = 0`：

  `STM32Cube_FW_L5_V1.0.0\STM32Cube_FW_L5_V1.0.0\Projects\STM32L552E-EV\Templates\TrustZoneDisabled\`

该应用笔记上面提到的参考项目 `TrustZone` 已启用，TrustZone 是通过置位选项字节的 `TZEN` 位来使能的。

该项目模板的 `readme` 文件描述如何配置选项字节以匹配代码；它提供了一个很好的模板，可用于学习一些重要的配置用例。

在首次学习体验之后，用户可以选择创建一个空项目，或者使用 STM32CubeMX 给自己的应用创建一个新项目，也可以两者都尝试一下。

固件 STM32Cube_FW_L5 包含许多面向不同外设的其他示例项目，带有 STM32CubeIDE 项目文件。可以将这些项目导入 STM32CubeIDE 并进行研究，以了解如何使用 STM32L5 外设。

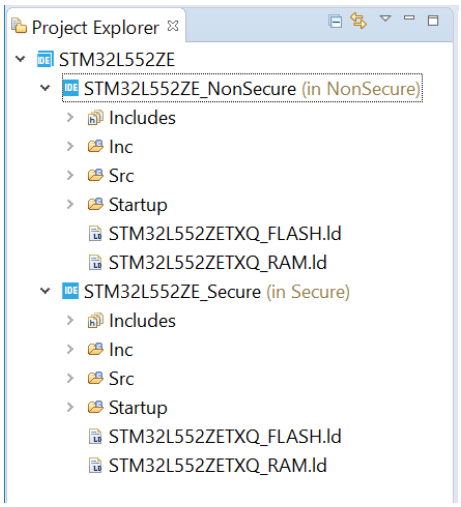
1.3 选项字节

若要详细了解选项字节有关信息，请参见 STM32L5 Series（RM0438）中的微控制器参考手册。对于作为此应用笔记基础的特定示例项目模板，示例项目的 `readme.txt` 文件中列出了正确的选项字节值。用户必须使用 STM32CubeProgrammer （STM32CubeProg）更改选项字节

1.4 特殊的分层项目结构面向安全多核 MCU

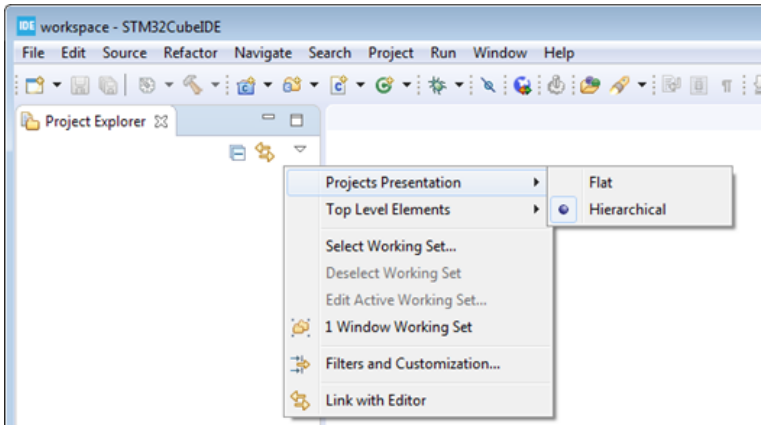
在导入或创建项目之前，考虑一些项目结构概念很重要。创建 STM32L5 项目之后，项目结构自动分层。单核项目的项目结构是扁平的。在多核项目或采用启用了 TrustZone®的 MCU 的项目（如 STM32L5 Series 中所示）中，使用分层项目结构。当用户创建或导入一个项目时，它由一个根项目和被称为 MCU 项目的子项目组成。MCU 项目是真实 CDT 项目；它们可以包含构建和调试配置，而根项目不能。根项目是一个简单的容器，允许在安全和非安全 MCU 项目之间共享公共代码（在 STM32L5 Series 的情况下），如图 1 中所示。

图 1. STM32L5 项目，采用分层项目结构



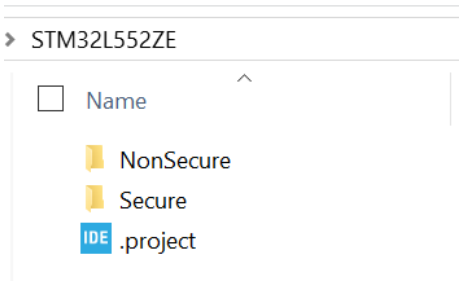
如果设置已更改，或者项目不在分层结构中，则可以进行更改，如图 2 中所示。

图 2. 在扁平和分层项目结构之间更改可视化表示形式



在文件系统中，两个 MCU 项目位于根项目中，根项目只包含一个.project 文件。

图 3. 根项目带有.project 文件



2 创建和导入项目

本章描述如何导入或创建基于 STM32L5 Series 的项目。首先解释如何导入 STM32CubeL5 MCU 开发包中可用的示例项目模板。在导入、构建、调试并向非安全可调用项添加一些函数调用之后，本文将说明如何创建自己的空项目，从示例项目中复制完全相同的资源作为后续应用程序的基础模板。

提示 *不建议在示例项目中继续应用程序开发，主要是因为项目中的所有资源都链接到项目中：*

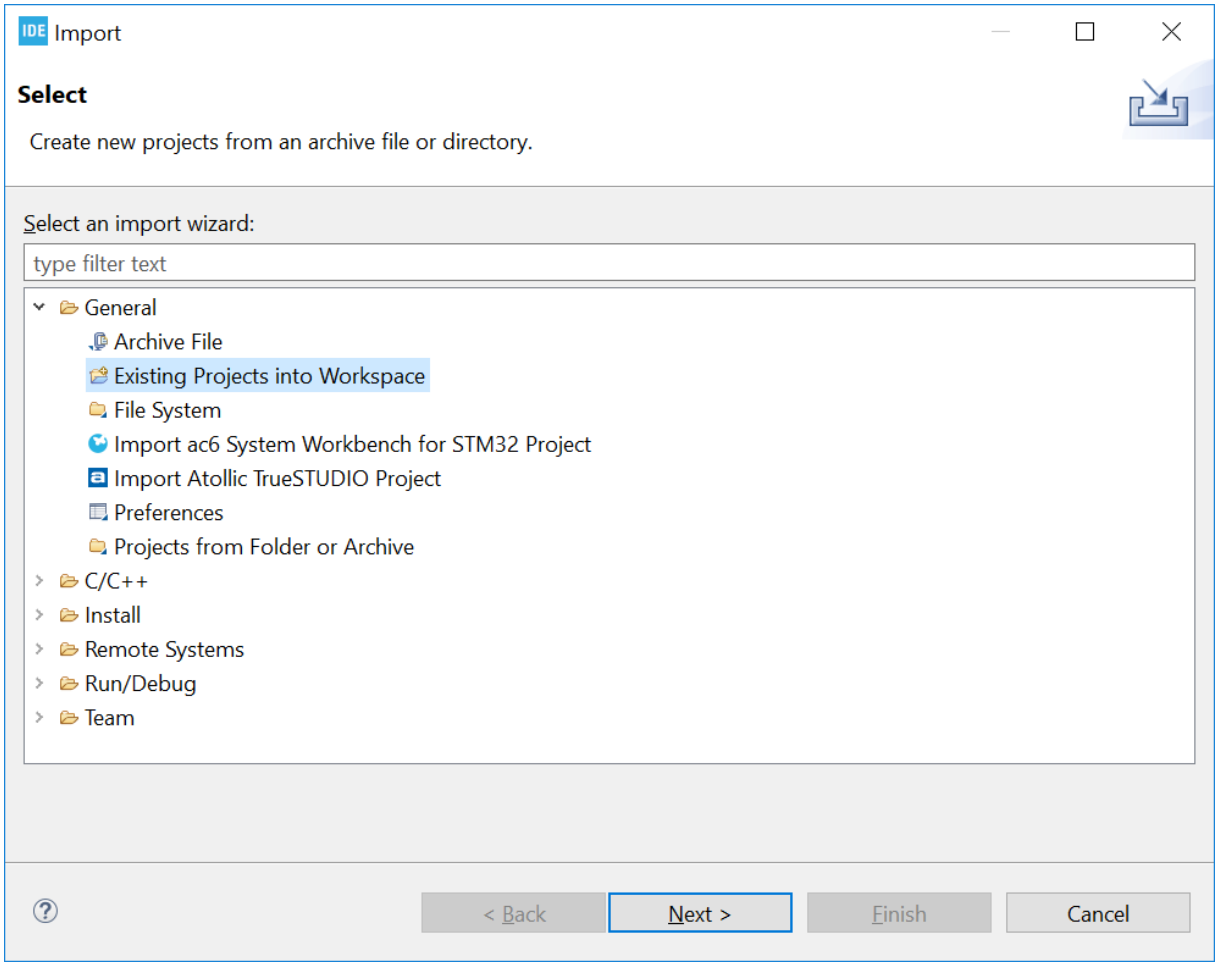
- *这意味着项目不是独立的，这使版本控制更加困难*
- *驱动程序资源与所有其他项目共享*
- *ECLIPSE™ CDT 索引器不能总是解析链接的资源，并正确启用所有代码导航和可视化特性*

使用示例项目作为模板创建新的空项目是后续应用程序开发的更好方法。

2.1 导入 TrustZone®项目模板 STM32CubeIDE

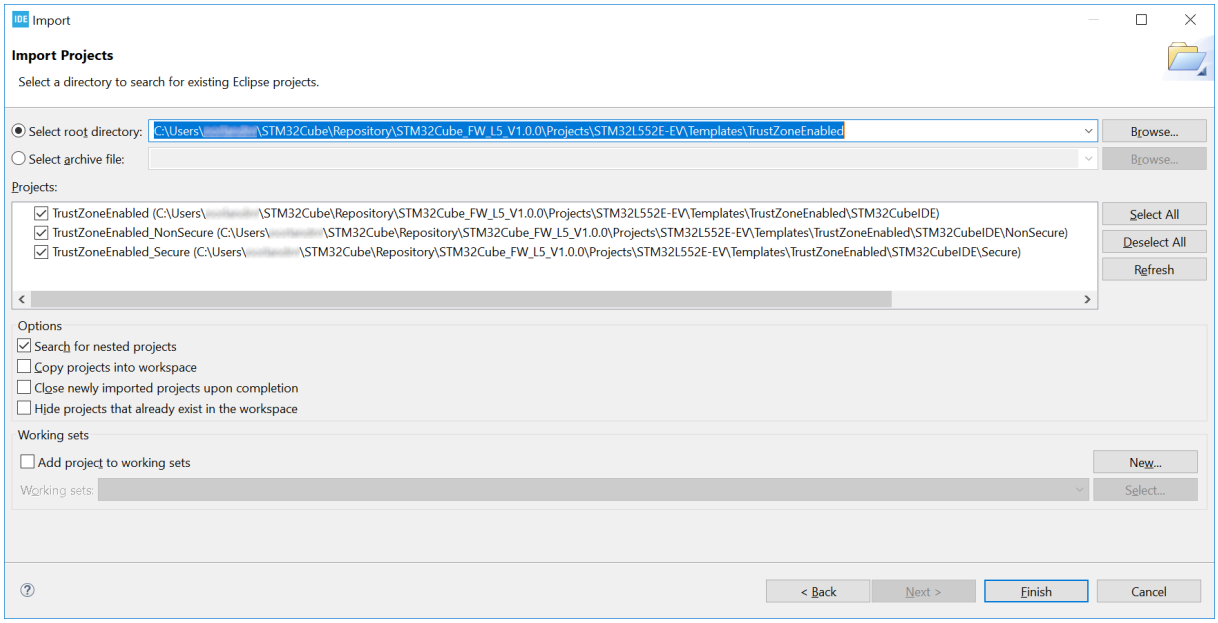
为了将 STM32CubeL5 模板项目导入 STM32CubeIDE 中，首先前往[File]>[Import]并选择 *Existing Projects into Workspace*，如图 4 中所示。

图 4. 导入项目模板



然后选择合适的项目，在 Windows®操作系统中默认位于 User（用户）文件夹中，例如$HOME\STM32Cube\Repository\STM32Cube_FW_L5_VX.X.X\Projects\STM32L552E-EV\Templates\TrustZoneEnabled\（请参见图 5）。

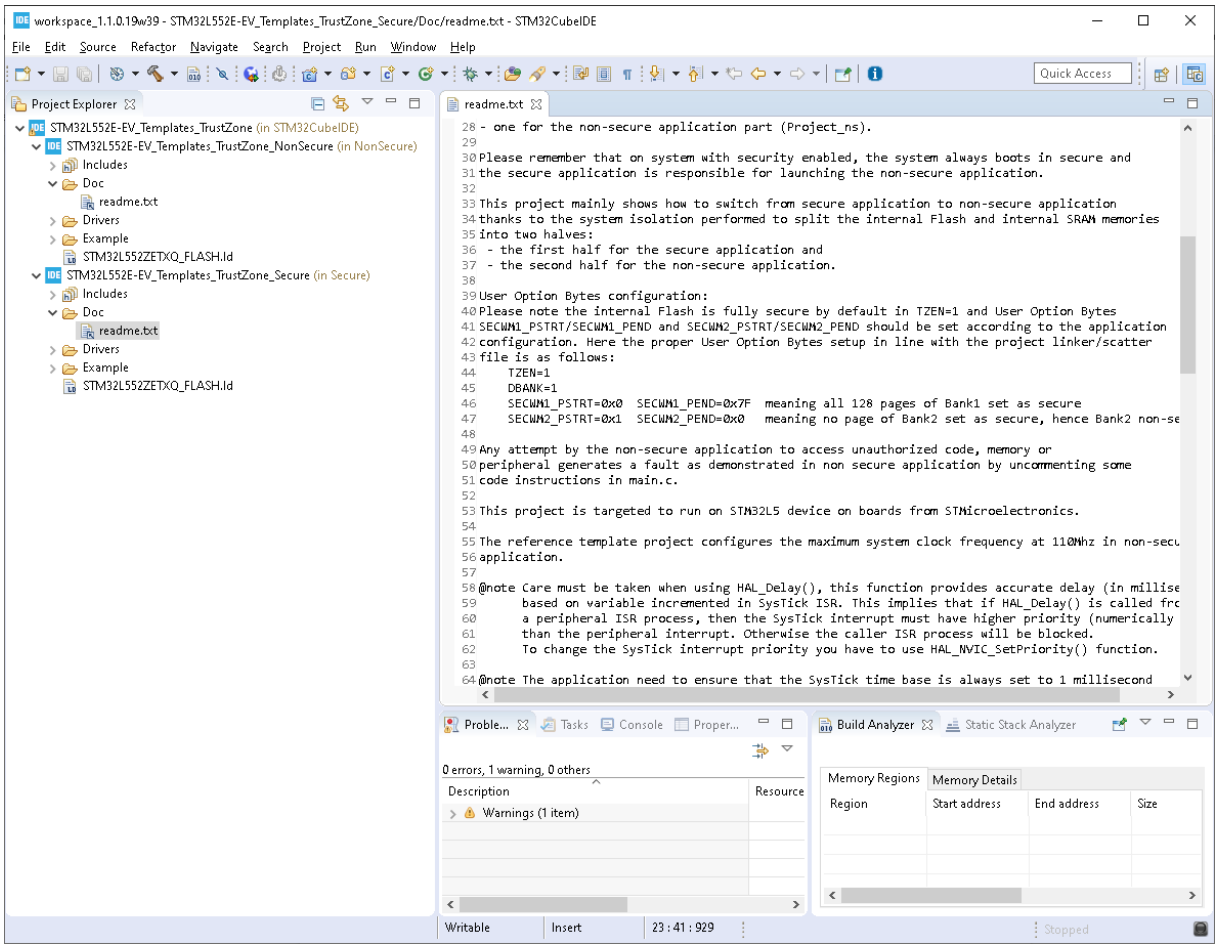
图 5. 选择 STM32L552E-EV 模板项目



Attention: *当从 STM32Cube MCU 开发包导入项目时，不要使用 Copy projects into workspace 设置，因为这样会打破与共享代码（比如 MCU 开发包中的 HAL 和 CMSIS 驱动）的链接。*

选择完所有三个项目之后，点击[Finish]以导入模板项目。

图 6. 构建导入的模板项目



2.2 探索示例项目

如要熟悉示例项目，首先阅读项目的 `readme.txt` 文件，该文件位于 `Doc` 文件夹。

2.2.1 选项字节

使用 STM32CubeProgrammer 根据 `readme.txt` 文件设置选项字节。对于启用了 TrustZone®项目，这些设置通常是：

- `TZEN = 1`
- `DBANK = 1`
- `SECWM1_PSTRT = 0x0` 且 `SECWM1_PEND = 0x7F`，意味着 Bank1 的 128 页都被设置为安全的
- `SECWM2_PSTRT = 0x1` 且 `SECWM2_PEND = 0x0`，意味着 Bank2 没有一页是安全的，因此 Bank2 是非安全的

重要说明：

- 在启用 `TZEN` 并应用到目标之前，`SECWMx` 选项字节不可见
- 始终仔细查看 `readme.txt` 文件，以确定正确的选项字节值

若需更多信息，请参见 STM32CubeProgrammer 用户手册（UM2237）。

2.2.2 探索链接脚本、内存分区和 SAU 初始化

每个安全的和非安全项目都是用其自身的链接脚本构建的。本节介绍在使用 STM32L552ZE 微控制器的情况下，安全链接脚本和非安全链接脚本的一些区别。

STM32L552ZE 的闪存大小是 512 KB。当选项字节 DBANK = 1 时，闪存被分成两个 256 KB 的存储块：一个是安全存储块，另一个是非安全存储块。然后，这两个存储块被进一步划分为面向不同类型工具链输出的区域。

安全闪存链接脚本：

```
MEMORY
{
  RAM     (xrw)   : ORIGIN = 0x30000000,  LENGTH = 96K
  ROM     (rx)    : ORIGIN = 0x0C000000,  LENGTH = 248K
  ROM_NSC (rx)    : ORIGIN = 0x0C03E000,  LENGTH = 8K    /* 非安全可调用区域 */
}
```

非安全闪存链接脚本：

```
MEMORY
{
  RAM     (xrw)   : ORIGIN = 0x20018000,  LENGTH = 96K
  ROM     (rx)    : ORIGIN = 0x8040000,   LENGTH = 256K
}
```

在本例中，安全应用程序从 0x0C00 0000 开始，而非安全应用程序从 0x0804 0000 开始。在安全链接脚本中，留出一个 8 KB 的区域来包含非安全调用项，它是允许非安全应用程序调用安全区域中定义的一组函数的粘合层。

链接脚本必须与内存分区头文件保持一致。在本例中，内存分区是在文件 partition_stm321552xx.h 中定义的。该头文件包含用于配置 SAU 的设置。下面是一个内存区域的摘要：

```
/* Initialize and enable the SAU */
#define SAU_INIT_CTRL          1
#define SAU_INIT_CTRL_ENABLE   1
...
/* <e>Initialize SAU Region 1 with memory attributes */
#define SAU_INIT_REGION1       1
#define SAU_INIT_START1        0x08040000      /* start address of SAU region 1 */
#define SAU_INIT_END1          0x0807FFFF      /* end address of SAU region 1 */
/* Region can be set as: 0 = non-secure, 1= secure, non-secure callable */
#define SAU_INIT_NSC1          0
```

共有八个内存区域可以配置为非安全或安全/非安全可调用。SAU 被配置为启动顺序的一部分：
Reset_Handler() → 调用 SystemInit() → 调用内联的 TZ_SAU_Setup() 函数。

TZ_SAU_Setup() 设置 SAU。

提示

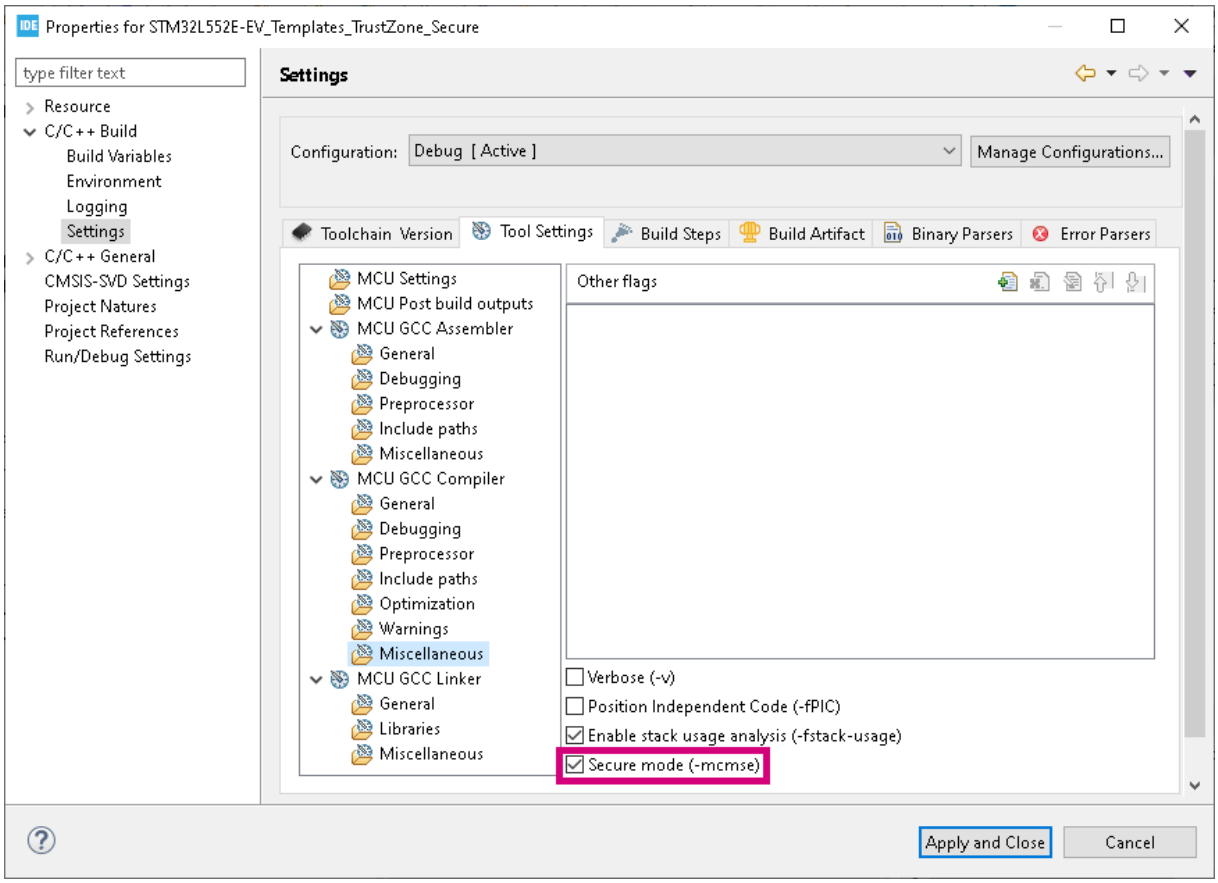
1. *链接脚本和分区头文件必须保持一致。如果非安全项目的链接脚本定义了用于非安全用途的某个闪存区域，那么分区头文件中也必须对该区域进行适当描述。在上面的示例中，链接脚本和分区头文件都指出 0x0804 0000 被用作非安全闪存区域。*
2. *在初始化 SAU 之前，CPU 无法访问非安全内存。CPU 或调试器读取非安全内存的任何尝试都会导致 RAZ（读取为零）：只返回零。必须执行 TZ_SAU_Setup() 函数才能访问非安全内存。*
3. *调试器能够通过使用虚拟配置配置 SAU 的方式编写非安全二进制代码。调试器在 Flash 加载完成后发出 reset 命令，允许使用通过用户的 SAU 设置进行初始化的 SAU 进行调试。*

2.2.3 TrustZone®- 相关构建设置

在启用了 TrustZone® 的 STM32L5 项目中，还配置了一些附加的编译设置。在典型用例中，建议用户不要更改这些设置。

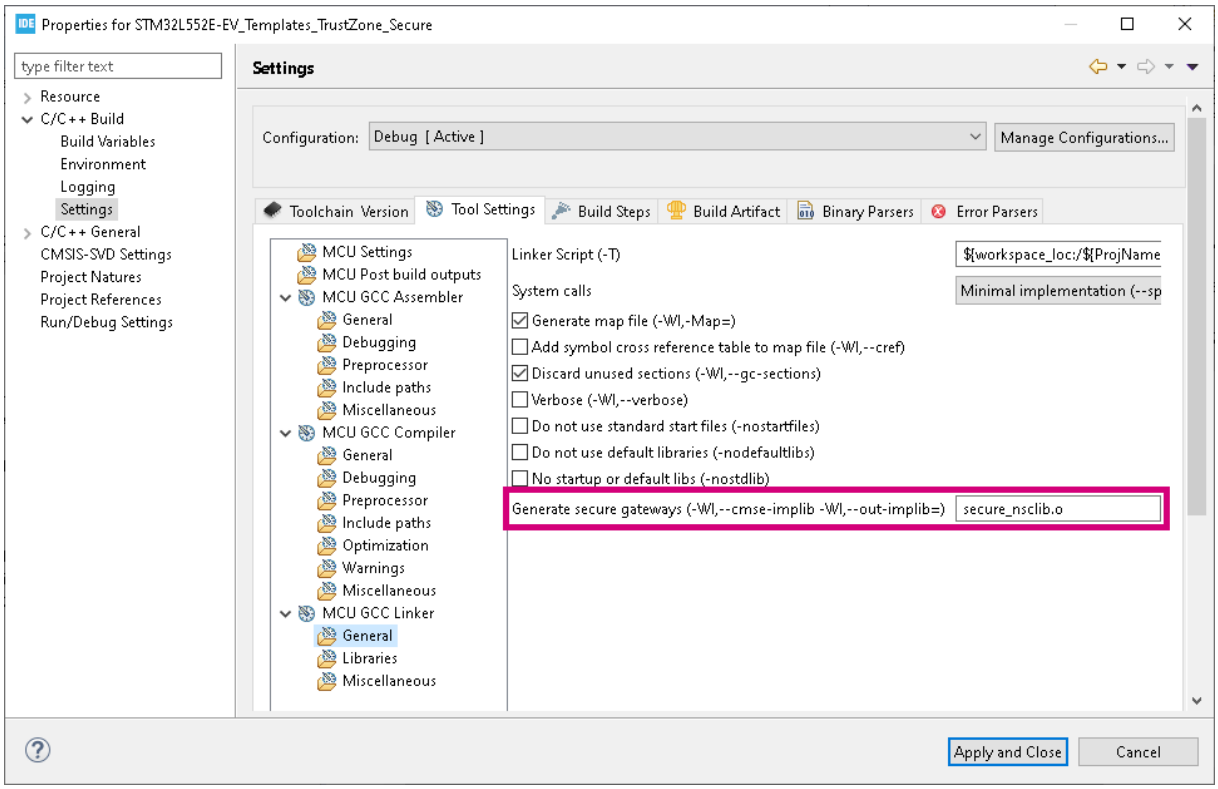
在安全项目中，使用-mcmse 属性调用编译器。

图 7. **STM32L5 安全项目：编译器调用**



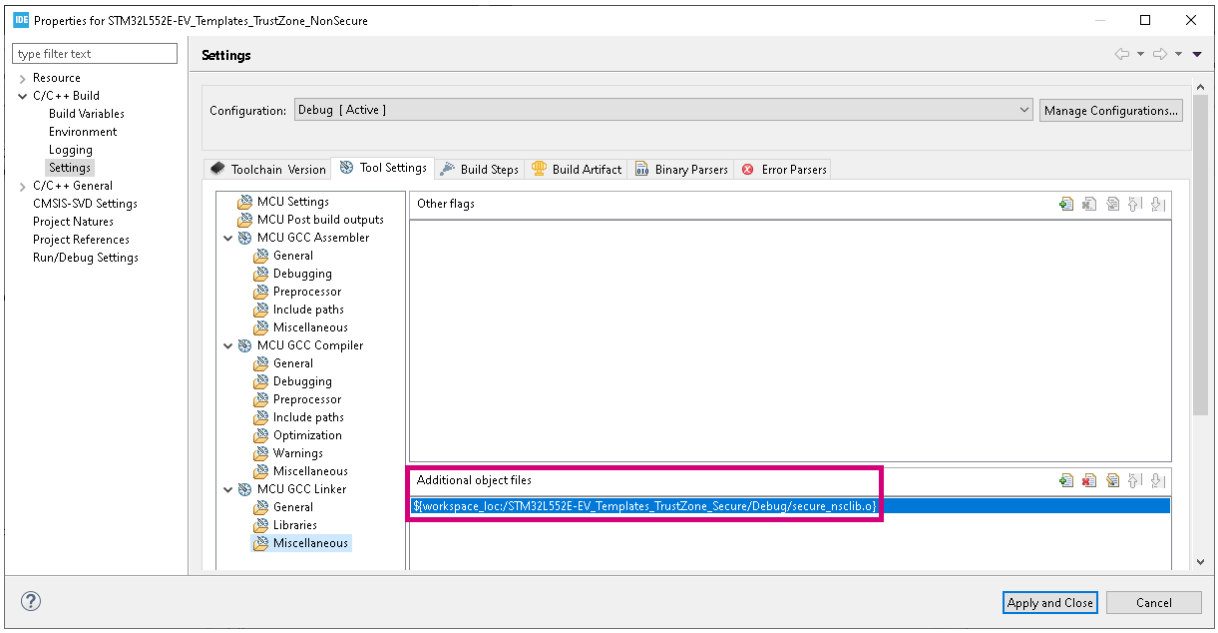
在安全项目中，对链接器进行配置，为非安全可调用项生成安全网关。

图 8. **STM32L5 安全项目：链接器配置**



在非安全项目中，对链接器进行配置，链接由安全应用程序构建的非安全可调用库中的对象。

图 9. **STM32L5 非安全项目：链接器配置**



该示例项目中不需要任何更改。非安全项目包括来自安全项目的安全库。因此，如果有必要，会在非安全项目之前扫描安全项目是否变更并重新编译。

用户检查点

此时，建议用户在继续下一步之前，对是否正确理解构建机制进行评估。尝试构建非安全项目，从而自动触发安全项目的构建。

2.2.4 RDP-level 0：加载和调试安全项目与非安全项目

可以使用下列任何工具将应用程序加载到 STM32L5 目标中：

- ST-LINK GDB 服务器，通过调用绑定的 STM32CubeProgrammerCLI 版本
- OpenOCD
- STM32CubeProgrammer 独立式

本应用笔记主要关注 ST-LINK GDB 服务器以及 OpenOCD 的使用。

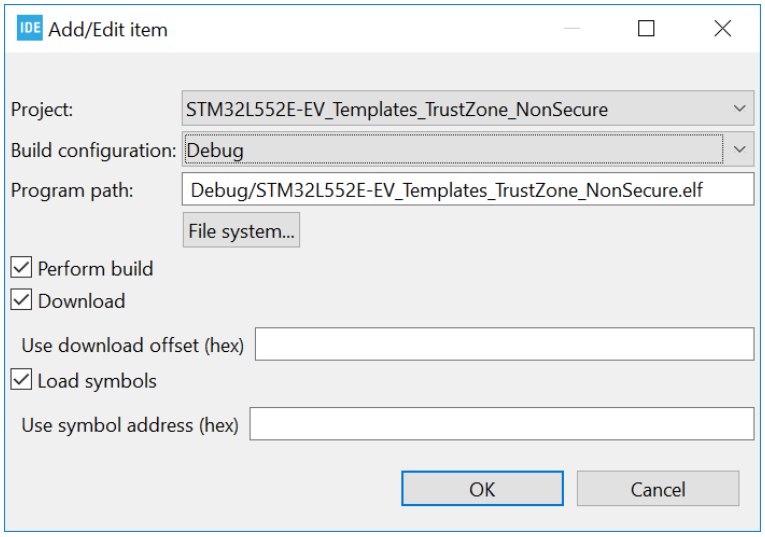
提示 *不能为根项目创建调试配置，只能为两个应用程序项目（安全和非安全）创建调试配置。*

在所考虑的用例中，假设用户具有对所有代码的完全访问权，并希望调试完整的应用程序。选项字节 RDP-level 必须仍然设为 0（0xAA）。

要创建调试配置，请执行以下步骤：

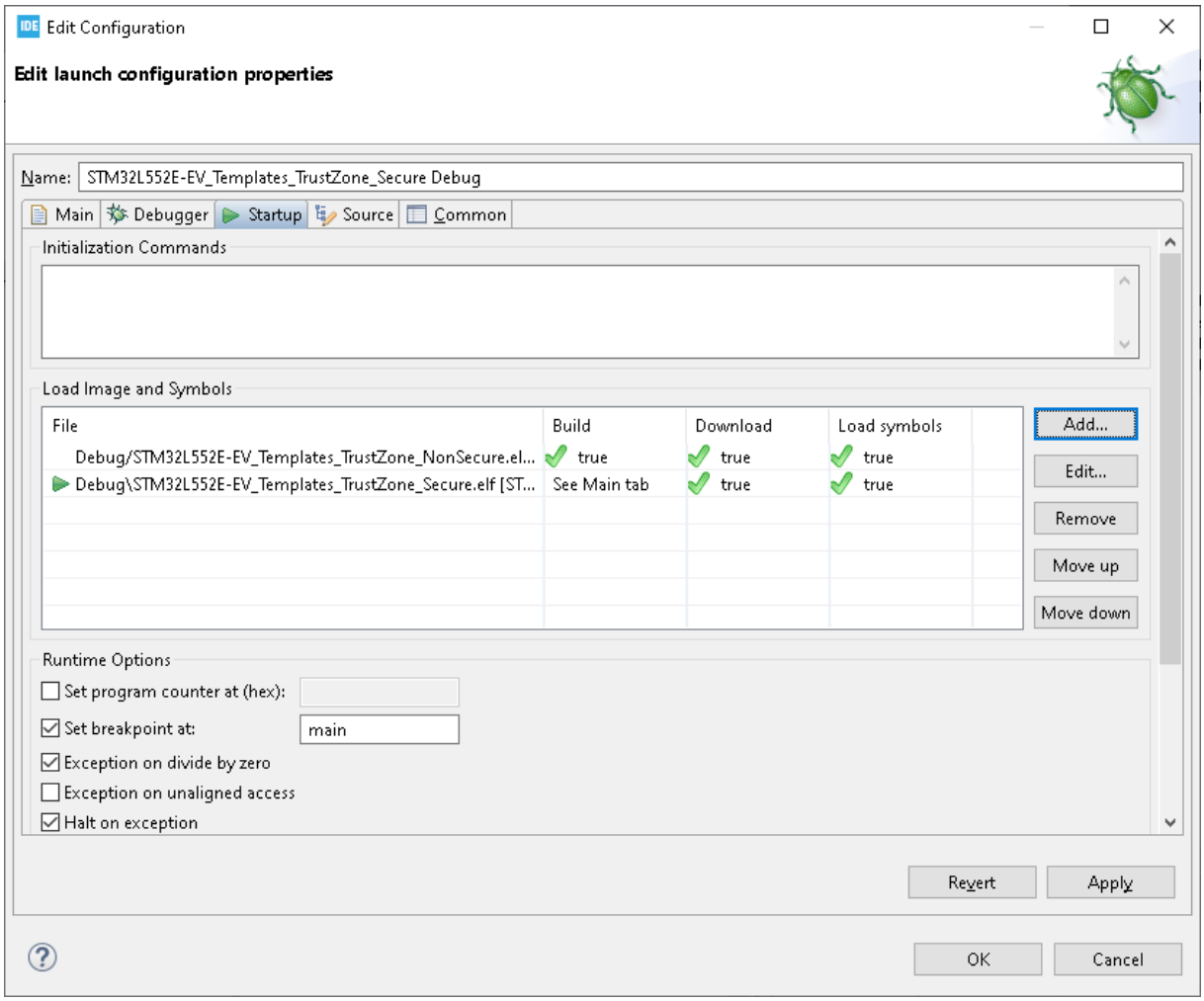
1. 在[Project Explorer（项目浏览器）]选择安全项目。
2. 右键单击[Debug As…]，然后选择[STM32 Cortex-M C/C++ Application]。
3. 前往 *Startup（启动）*选项卡。在 *Load Image and Symbol（加载图像和符号）*表中，目前只添加安全二进制文件。为了加载和调试安全和非安全二进制文件，用户必须手动将非安全二进制文件添加到加载列表中。
4. 点击[“添加”]。
   a. *Project（项目）*：选择非安全项目
   b. *Build Configuration（构建配置）*：调试
   c. 确保勾选了 *Download（下载）*和 *Load symbols（加载符号）*复选框

图 10. 添加二进制文件和加载符号到非安全项目



将非安全 elf 文件添加为要加载到嵌入式目标的 elf 文件列表的一部分后，加载列表如图 11 中所示。

图 11. 启动配置加载列表



Attention: *STM32L5 Series TrustZone®启用后，器件始终在安全状态下启动。调试器使用 Load image and Symbols（加载映像和符号）表中最后一个映像的信息设置程序计数器。确保安全映像位于加载列表的底部。*

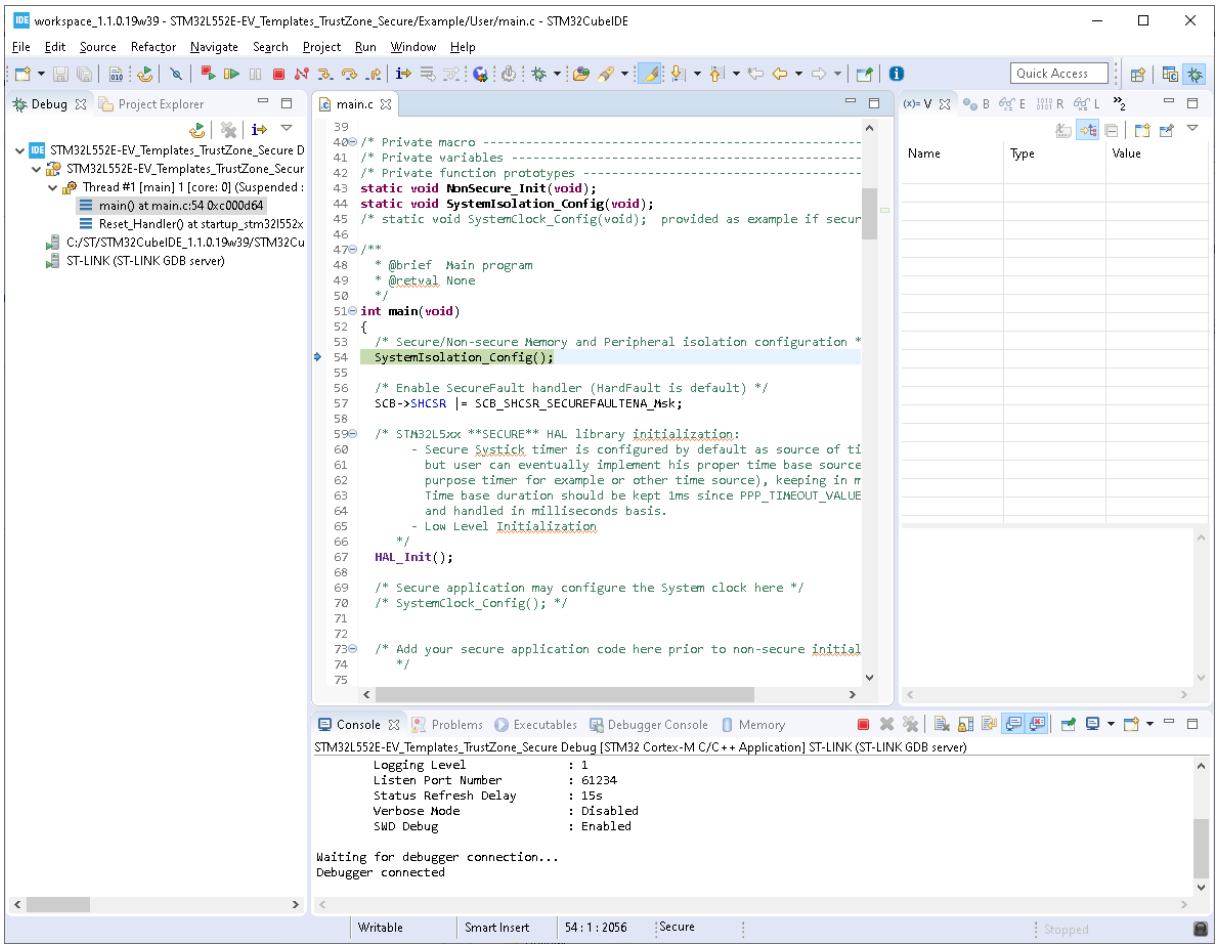
提示 *在启动调试会话之前，STM32CubeIDE 检查某些代码更改是否需要进行新的构建。如果采用大型代码库，这样的检查所需时间将增加。一些用户可能希望禁用此检查，以实现更快的调试启动过程。在这种情况下，他们还必须跟踪构建变更，并确保手动构建。*

上面介绍的调试配置设置可以确保自上次构建以来代码中的任何更改都会触发新的构建。下载两个映像，GDB 从两个二进制文件加载符号，以便能够将指令映射到 C 代码。

假设 STM32L5 器件已连接，且选项字节已正确配置，点击[OK（确定）]启动第一个调试会话。

应用程序在安全应用程序的 `main()` 中的第一行停止。

图 12. 调试器在安全主函数上停止



在 `main.c:54 - SystemIsolation_Config()` 中：在该函数中，安全端读取内存块和外设，以供非安全端访问。

使用单步执行功能遍历安全应用程序的第一行，并了解如何配置器件，以便从安全上下文跳转到非安全上下文。

在 `main.c` 中：有一个对 `NonSecure_Init()` 的函数调用。单步执行该函数，如图 13 中所示。

图 13. 将安全跳转初始化为非安全跳转

```
132   */
133 static void NonSecure_Init(void)
134 {
135   funcptr_NS NonSecure_ResetHandler;
136
137   SCB_NS->VTOR = VTOR_TABLE_NS_START_ADDR;
138
139   /* Set non-secure main stack (MSP_NS) */
140   __TZ_set_MSP_NS((*(uint32_t *)VTOR_TABLE_NS_START_ADDR));
141
142   /* Get non-secure reset handler */
143   NonSecure_ResetHandler = (funcptr_NS)(*((uint32_t *)((VTOR_TABLE_NS_START_ADDR) + 4U)));
144
145   /* Start non-secure state software application */
146   NonSecure_ResetHandler();
147 }
```

- `main.c:137`：使用非安全应用程序的中断向量的地址初始化非安全向量表偏移寄存器。
- `main.c:140`：初始化非安全主堆栈指针。
- `main.c:143`：从非安全中断向量表中获取非安全复位处理程序的地址。这是表中的第二个条目。
- `main.c:146`：执行函数指针，以跳转到非安全复位处理程序。

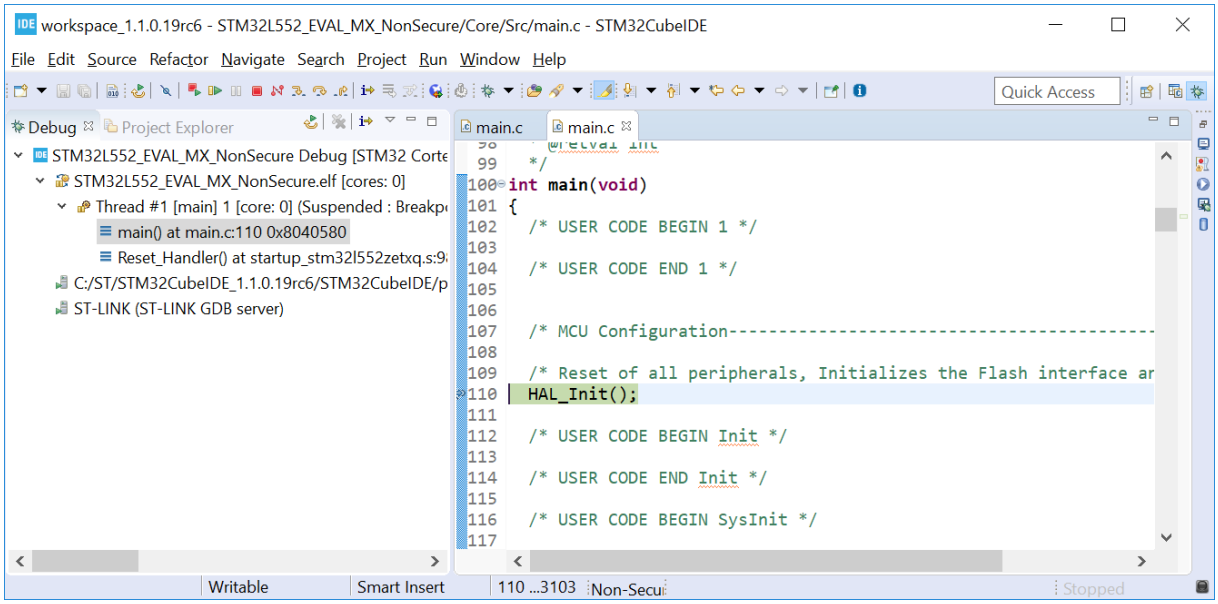
函数指针机制是从安全上下文跳转到非安全上下文的唯一方法。要了解更多信息，请查看安全项目中的 `main.h` 文件。它包含以下各行：

```
# define CMSE_NS_CALL  __attribute((cmse_nonsecure_call)) /* Function pointer declaration in non-secure */
typedef void CMSE_NS_CALL (*funcptr)(void);
typedef funcptr funcptr_NS; /* typedef for non-secure callback functions */
```

提示 *main 函数中的断点仅为安全应用程序而设置。因此，为了在非安全 main 程序的第一行停止，用户必须手动设置断点。*

在 `main.c:146` 上执行完 ***Step into***（*步进*）操作后，或者用户在非安全 `main()` 中手动设置断点，然后按下 ***Continue operation***（*继续操作*），执行将在非安全 `main()` 中的第一行停止，如 图 14 中所示。

图 14. 从安全跳转到非安全



2.2.5 RDP-level 0.5：加载和调试非安全项目

在 RDP-level 0.5 中，调试器无法读取或写入与安全端相关的任何信息。因此，STM32L5 Series 器件必须已经包含一个安全映像，该映像正确初始化 SAU，以便能调试非安全项目。此外，非安全链接脚本和 SAU 设置必须同步。本应用笔记中使用的示例项目可作为参考。

当前章节中的描述假设安全端已经编程并且 RDP-level 选项字节设置为 0.5。

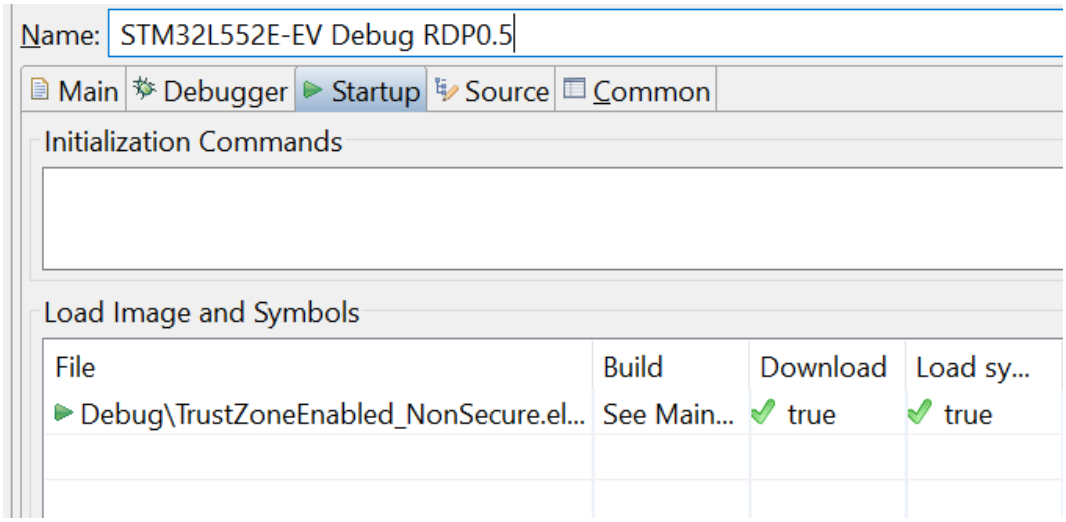
调试配置

如要创建调试配置以加载非安全映像，请执行以下步骤：

1. 在[Project Explorer（项目浏览器）]中选择非安全项目。
2. 右键单击[Debug As...]，然后选择[STM32 Cortex-M C/C++ Application]。
3. 为配置指定一个有用的名称，以便易于识别。在第 2.2.5 节 中的案例中，添加了后缀 *RDP0.5*。

4. 前往 *Startup（启动）*选项卡。验证非安全项目 `elf` 文件的 *Download（下载）*和 *Load（加载）符号*设为 true（真）。

图 15. RDP 0.5 的启动配置加载列表



点击[OK（确定）]开始调试配置。器件执行重置，调试器等待 CPU 从安全上下文跳转到非安全上下文，然后它才能停止执行。

提示

在 RDP-level 0.5 中，如果用户试图在 CPU 执行安全代码时停止 CPU，则停止事件将一直等待，直到 CPU 返回非安全端。如果 CPU 在安全端耗时超过两秒，则停止操作处于超时状态。

在“Debug Configurations（调试配置）”对话框中，“Startup（启动）”选项卡包含一个“Max halt timeout（最大停止超时）”选项，可以进行配置以允许 ST-LINK GDB 服务器等待更长的超时时间。如要允许 GDB 也等待更长的超时时间，需要创建 `.gdbinit` 文件。该文件必须在 `PROJECT_ROOT/.gdbinit` 中可用，并包含以下命令到 `gdb` .中，其中的值以秒为单位，可根据应用需要进行更改：

- `set remotetimeout 50`
- `set tcp connect-timeout 50`

2.2.6 TrustZone® *Debug* 透视图和视图中的特定扩展

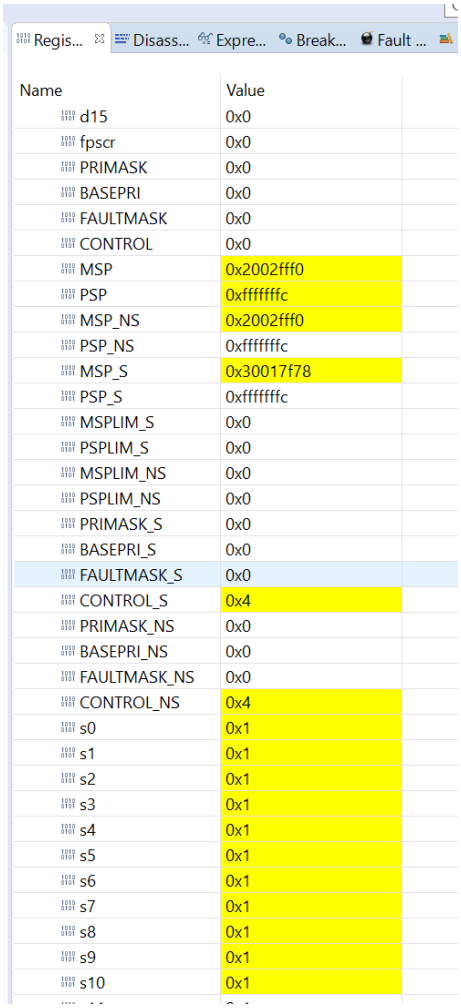
STM32CubeIDE 中有特定功能，以支持启用了 TrustZone®的 STM32 器件。

*Debug（调试）*透视图的底部有一个 *Security（安全）*指示器。该指示器根据应用程序执行停止的上下文（安全或非安全）来显示 *Secure（安全）*或 *Non-Secure（非安全）*状态。该指示器的两种状态显示在图 16 中。

图 16. *Security（安全）*指示器

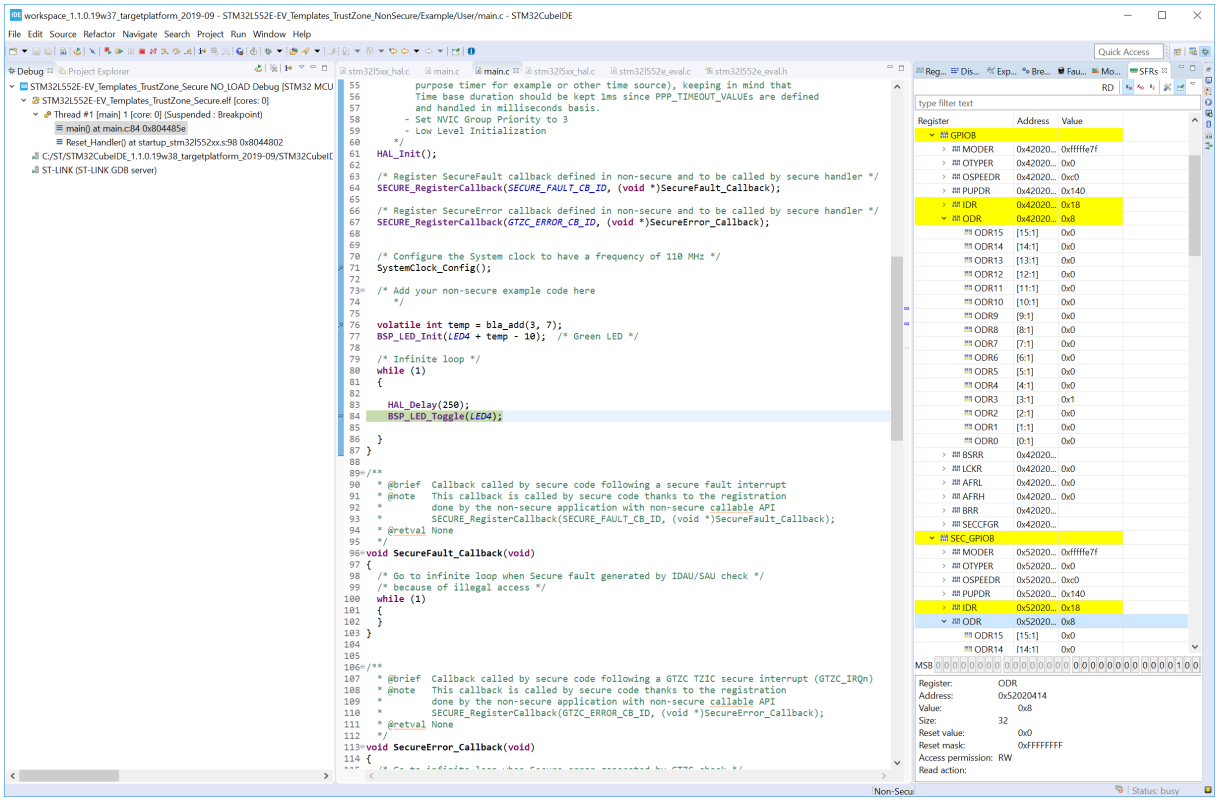
扩展 *Registers（寄存器）*视图以显示已存储的安全和非安全寄存器。如果这些寄存器是安全的，则加后缀 *_S*；如果是非安全的，则加后缀 *_NS*。

图 17. 已存储的安全和非安全寄存器



SFR 视图显示文件 CMSIS-SVD 的内容。对于可以在安全和非安全上下文之间共享的每个外设，此文件包含非安全和安全内存地址。

图 18. SFR 视图显示带有安全和非安全地址空间的外设

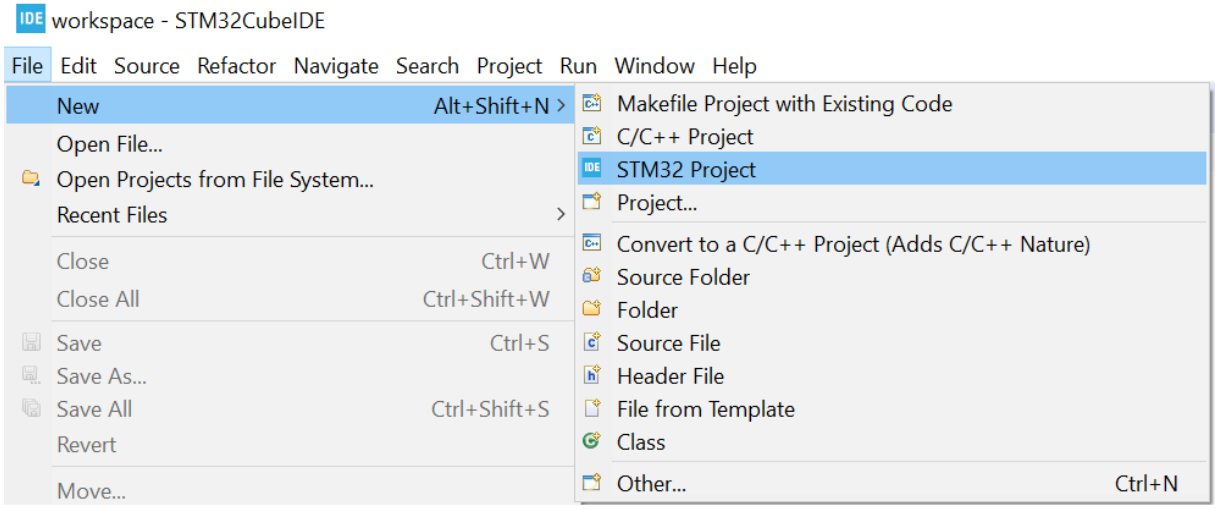


Fault Analyzer（*故障分析器*）视图也相应更新。它显示适用于 STM32L5 器件的新异常类型，并根据故障情况和 FPU 使用情况计算异常堆栈帧。

2.3 创建启用了 TrustZone®的空项目

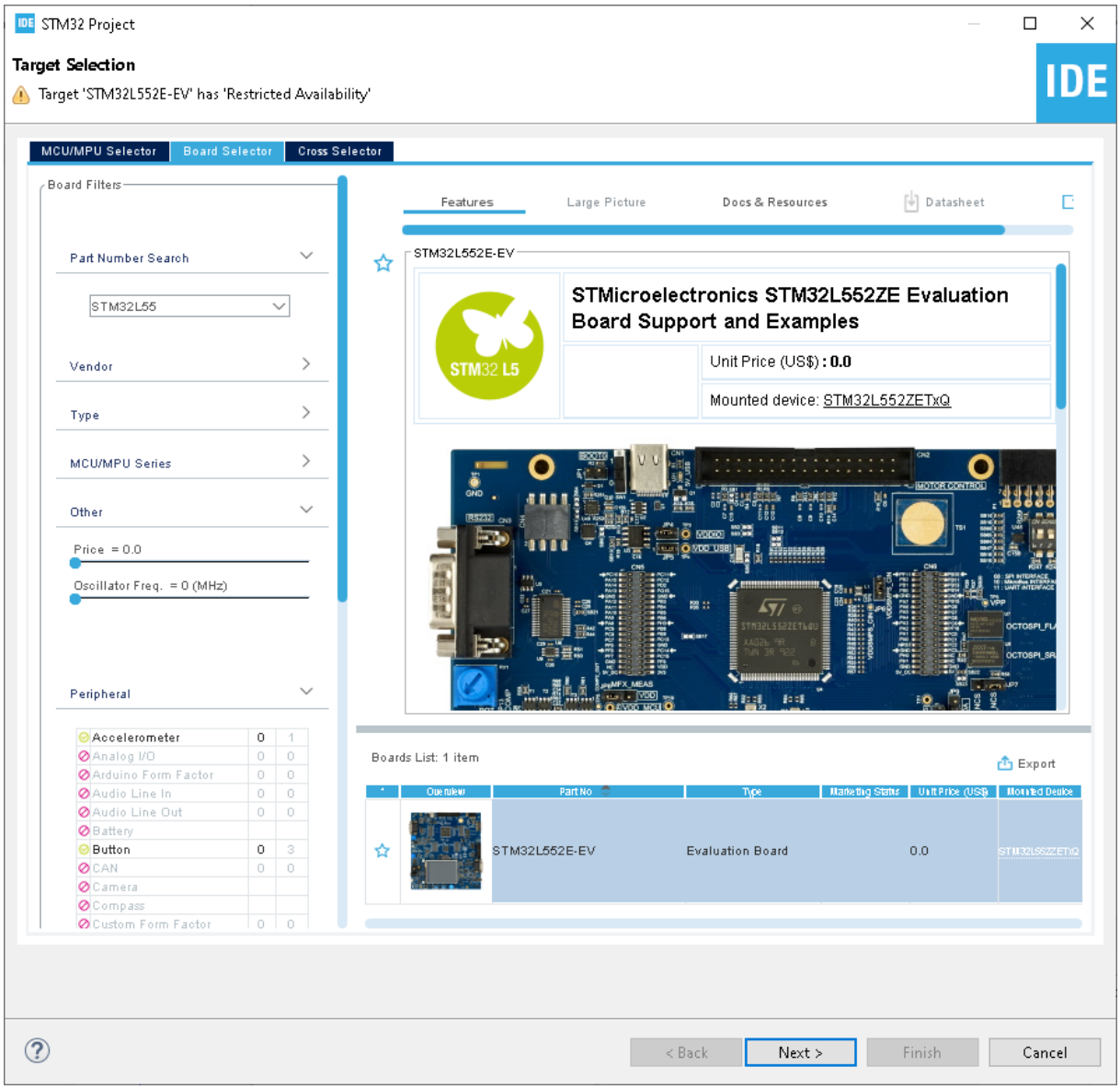
本章假设读者熟悉第 2.1 节 导入 TrustZone 项目模板 STM32CubeIDE 并了解选项字节、内存分区、构建和调试。若需开始一个新项目，转至[File（文件）]>[New（新建）]>[STM32 project（STM32 项目）]。

图 19. 新建空项目



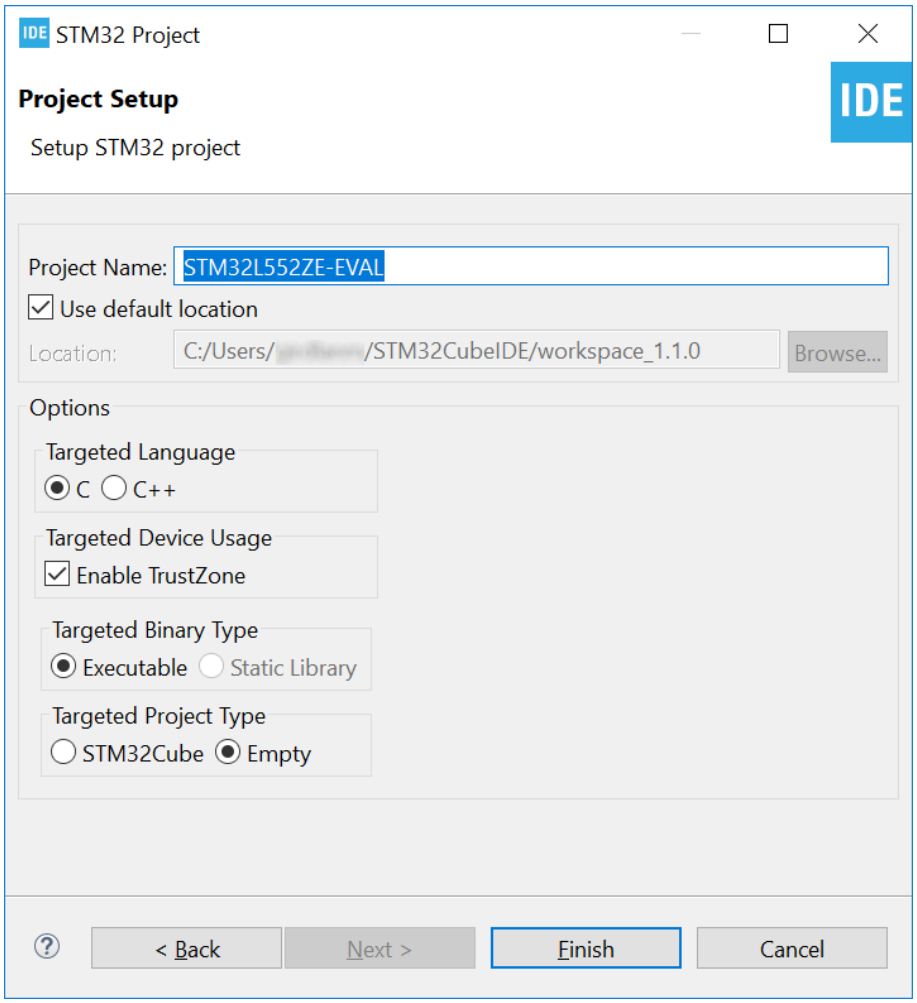
选择 MCU 或板。在本例中，选择了 STM32L552E-EV 评估板。点击[Next（下一步）]。

图 20. 目标选择



选择了 MCU 或板之后，下一步是 *Project Setup（项目设置）*步骤。

图 21. 项目设置 - 选择空项目



- 命名项目。
- 确保选中 *Targeted Device Usage* 下的 *Enable TrustZone*。

  这将确保项目生成之后，三个项目位于分层结构中，而不是位于单一扁平项目中。根项目没有 CDT 本质，因此不了解构建配置或调试配置之类的概念。根项目只是面向两个目标项目的一个容器，允许在两个 MCU 项目之间共享一些通用代码。
  - 第一个 MCU 项目与应用的非安全部分相关。从这个意义上说，它类似于传统的 STM32 项目。
  - 第二个子项目与应用的安全部分相关。该项目将编译器配置为使用 `-mcmse` 标志进行编译，并使用链接器为非安全可调用函数生成对象文件。

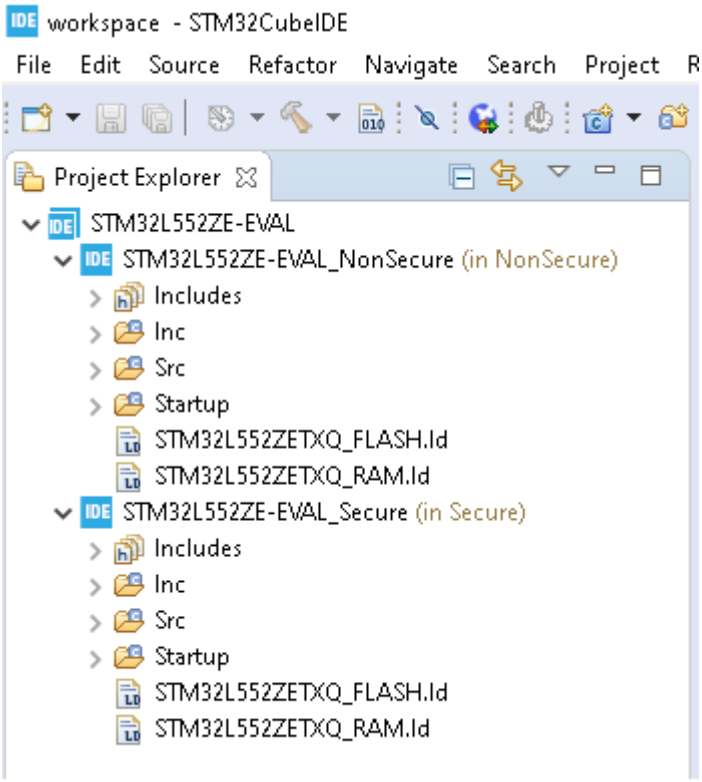
  如果未选中 *Enable Trust Zone*，从安全角度看，STM32L5 就像任何其他单核 STM32 微处理器一样工作。这不是主要用例，因此没有在应用笔记中记录。
- *Targeted Project Type（目标项目类型）*：选择 *Empty（空项目）*。这将生成一个空的项目框架，可以手动添加文件到该项目中。

提示

1. *“Enable TrustZone（启用 TrustZone）”选项是不可逆的。其结果有两种：一种是扁平项目结构（TrustZone® 被禁用），另一种是分层项目结构（TrustZone®启用）。一旦创建了项目，就不能在这两个项目结构之间切换。*
2. *在项目向导中启用 TrustZone®并不会启用选项字节 `TZEN`，后者必须使用 STM32CubeProgrammer 进行设置。*

在命名项目、勾选 *Enable Trust Zone*、设置 *Targeted Project Type* 为 *Empty* 之后，单击[Finish]。空项目结构创建完毕，如图 22 中所示。

图 22. 空项目结构



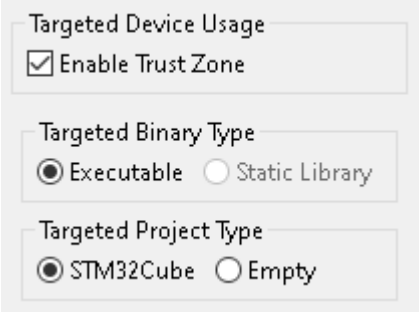
一个根项目包含两个子项目。只能构建和调试两个 MCU 子项目。根项目只是一个容器。安全 `main()` 函数必须用代码进行更新，以调用非安全应用程序。如何做到这一点可以在安全模板项目中、其他安全固件示例项目中、或通过创建一个启用了 TrustZone®的 STM32CubeMX 项目来学习研究。

第 2.2.4 节 RDP-level 0：加载和调试安全项目与非安全项目或第 2.2.5 节 RDP-level 0.5：加载和调试非安全项目中描述了如何调试这些项目。

2.4 创建一个启用了 TrustZone®的 STM32CubeMX 项目

在第 2.3 节 创建启用了 TrustZone 的空项目中执行与创建空项目相同的步骤，以创建一个带有 `.ioc` 文件的项目，其中的资源由 STM32CubeMX 控制。记得要勾选 *Enable Trust Zone* 复选框，并将 *Targeted Project Type* 设为 *STM32Cube*。

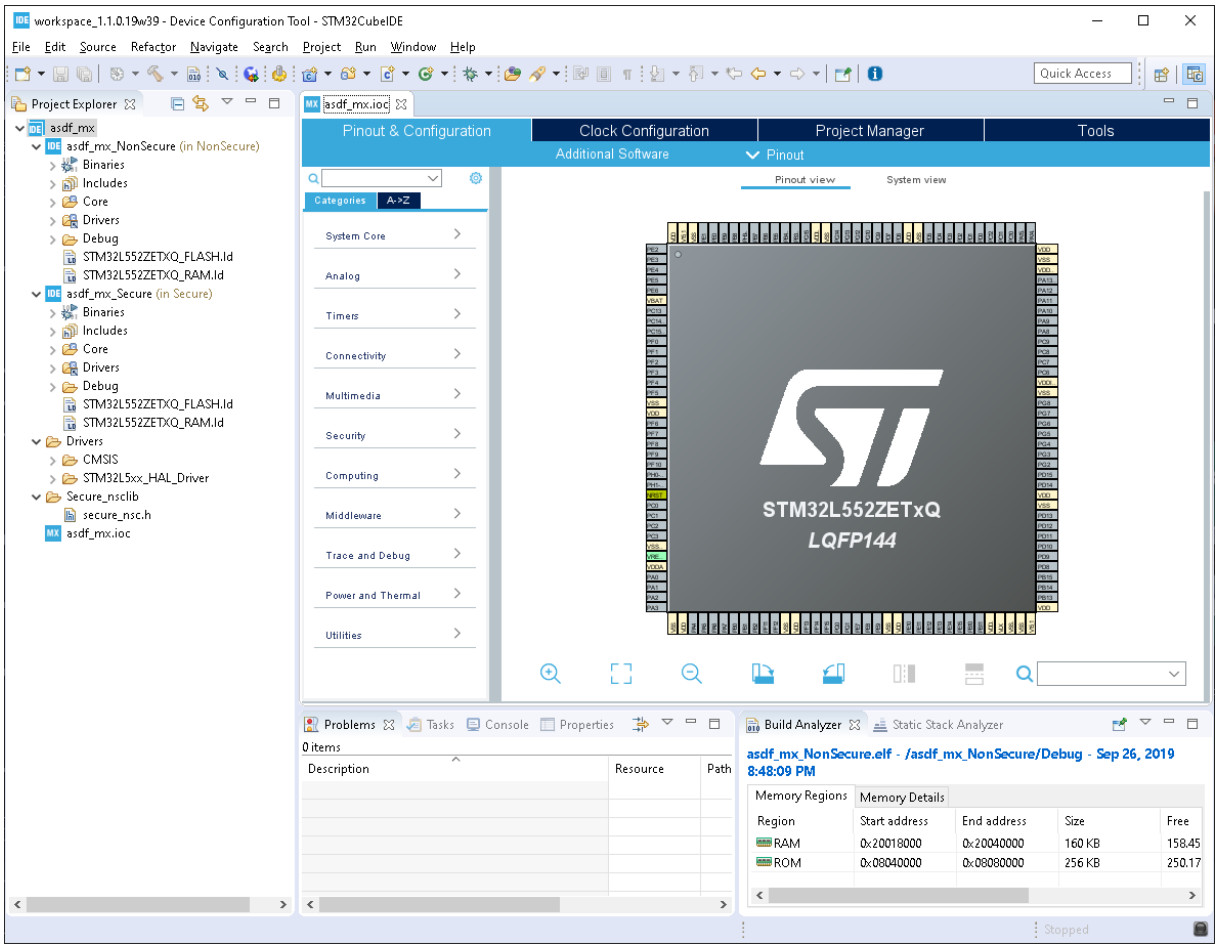
图 23. 项目设置 - 选择 STM32Cube



[File]>[New STM32 Project]>[Project Setup]页面，这是项目向导中的第二步。

图 24 显示由 STM32CubeIDE 中集成的 STM32CubeMX 生成的 STM32L5 项目截图。

图 24. 生成的项目来自 STM32CubeMX



包含 CMSIS 和 HAL 代码的 `Drivers`（驱动程序）文件夹存储在根项目中，并创建到其代码的链接，以便在两个 MCU 项目中构建代码。

第 2.2.4 节 RDP-level 0：加载和调试安全项目与非安全项目或第 2.2.5 节 RDP-level 0.5：加载和调试非安全项目中描述了如何调试这些项目。

3 从非安全域调用到安全域

本节说明如何从非安全应用程序将函数调用到安全应用程序。这是通过 GCC pragmas 和一个称为“non-secure callable（缩写为 NSC）”的粘合层完成的。

为了非安全函数能够访问安全函数，安全函数必须通过__attribute((cmse_nonsecure_entry))进行定义，如下面的例子所示：

```
uint32_t __attribute((cmse_nonsecure_entry)) getSecureKey(void)
{
    return 0xdeadbeef;
}
```

Important: *相应头文件中的函数原型不得使用 uint32_t __attribute((cmse_nonsecure_entry))。*

快速尝试这个例子：

- 将上面的代码片段复制粘贴到安全应用程序 main.c 中。将代码片段置于 USER CODE BEGIN PV Begin/End 部分或类似位置。
- 在根项目中，打开 Secure_nsclib/secure_nsc.h 头文件。添加下面的代码行到 secure_nsc.h：

  ```
  uint32_t getSecureKey(void);
  ```

  该头文件是一个实用的示例，因为它同时包含在安全和非安全项目中。
- 在非安全 main.c 中添加调用：

  ```
  int temp = getSecureKey();
  ```
- 构建非安全项目。这将触发安全项目的生成。

  安全项目的构建会生成 secure_nsclib.o，由非安全应用程序链接，以允许“非安全到安全”事务。
- 在包含 int temp = getSecureKey();的代码行放一个断点。
- 当达到断点时，启用指令步进模式。使用 ***step into***。
  - 在非安全代码中生成了一个单板函数_getSecureKey_veneer，用于处理非安全内存地址和安全内存地址之间的长跳转。这与安全无关，而是长跳转的结果。
  - 下一步是在安全内存中执行 sg（安全网关）指令，它授权“非安全到安全”事务。
  - 授权之后，分支执行安全内存中的 getSecureKey()函数。
  - 此函数可能调用另一个不是非安全可调用的安全函数。
    - 在 RDP-level 0 中，指令步进也可以在该函数中继续。
    - 在 RDP-level 0.5 中，安全应用程序中不允许步进。
  - 在 getSecureKey()的头域中，所有相关寄存器都被从安全端清除，将信息泄漏到非安全端。
  - 非安全调用结束后，执行返回到非安全端。

4 FAQ

4.1 调试器在加载 RDP0 中的非安全映像和安全映像之后崩溃

答案 #1

请仔细检查安全映像是否位于调试配置中加载列表的底部。此列表中的最后一个映像用于设置程序计数器启动地址，该地址必须在安全内存中。请参见第 2.2.4 节 RDP-level 0：加载和调试安全项目与非安全项目。

答案 #2

在 RDP-level 0.5 中，如果应用程序在安全上下文中耗时超过两秒，调试器就会超时。尝试扩展超时，如下所述。

在*“Debug Configurations（调试配置）”*对话框中，*“Startup（启动）”*选项卡包含一个*“Max halt timeout（最大停止超时）”*选项，可以进行配置以允许 ST-LINK GDB 服务器等待更长的超时时间。还必须指示 GDB 客户端使用更长的超时时间值，这是通过在项目根文件夹中创建一个名为`.gdbinit`的文件来完成的。该文件必须包含两行代码：

- `set remotetimeout 50`
- `set tcp connect-timeout 50`

请参见第 2.2.5 节 RDP-level 0.5：加载和调试非安全项目。

4.2 我在调试过程中的不同时间得到安全 GTZC 中断

如果出现任何 GTZC/TZIC 非法访问标志并且相应的中断已启用，则应用程序跳转到 GTZC 中断例程。可能会出现 GTZC/TZIC 非法访问标志，因为调试工具（ST-LINK GDB 服务器）试图在正确初始化 SAU 之前访问非安全内存地址。

例如，STM32CubeIDE 和其他 IDE 的以下特性触发内存读取：

- 在内存地址上设置断点将触发对该地址的读取。
- 使用 *Memory Browser*、*Expressions* 或其他视图从内存中读取数据，在出现停止事件或其他不透明 IDE 事件时触发读取。

可能的变通方法是：

- 不要在调试版本上启用 GTZC 中断。
- 在启用 GTZC 中断之前，在 SAU 初始化之后清除必要的 GTZC/TZIC 非法访问标志。

版本历史

表 1. 文档版本历史

日期	版本	变更
2020 年 1 月 9 日	1	初始版本。
2020 年 2 月 12 日	2	更新了第 4.2 节 我在调试过程中的不同时间得到安全 GTZC 中断。

目录

1 概述 2
1.1 先决条件 2
1.2 本文档中的用例 2
1.3 选项字节 2
1.4 特殊的分层项目结构面向安全多核 MCU 3
2 创建和导入项目 5
2.1 导入 TrustZone®项目模板 STM32CubeIDE 5
2.2 探索示例项目 7
2.2.1 选项字节 7
2.2.2 探索链接脚本、内存分区和 SAU 初始化 8
2.2.3 TrustZone®- 相关构建设置 8
2.2.4 RDP-level 0：加载和调试安全项目与非安全项目 11
2.2.5 RDP-level 0.5：加载和调试非安全项目 14
2.2.6 TrustZone® *Debug* 透视图和视图中的特定扩展 15
2.3 创建启用了 TrustZone®的空项目 17
2.4 创建一个启用了 TrustZone®的 STM32CubeMX 项目 20
3 从非安全域调用到安全域 22
4 FAQ 23
4.1 调试器在加载 RDP0 中的非安全映像和安全映像之后崩溃 23
4.2 我在调试过程中的不同时间得到安全 GTZC 中断 23
Revision history 24
目录 25
表一览 26
图一览 27

表一览

表 1. 文档版本历史 24

图一览

图 1. STM32L5 项目，采用分层项目结构 . . . 3
图 2. 在扁平和分层项目结构之间更改可视化表示形式 . . . 3
图 3. 根项目带有.project 文件 . . . 4
图 4. 导入项目模板 . . . 5
图 5. 选择 STM32L552E-EV 模板项目 . . . 6
图 6. 构建导入的模板项目 . . . 7
图 7. STM32L5 安全项目：编译器调用 . . . 9
图 8. STM32L5 安全项目：链接器配置 . . . 10
图 9. STM32L5 非安全项目：链接器配置 . . . 10
图 10. 添加二进制文件和加载符号到非安全项目 . . . 11
图 11. 启动配置加载列表 . . . 12
图 12. 调试器在安全主函数上停止 . . . 13
图 13. 将安全跳转初始化为非安全跳转 . . . 13
图 14. 从安全跳转到非安全 . . . 14
图 15. RDP 0.5 的启动配置加载列表 . . . 15
图 16. *Security*（安全）指示器 . . . 15
图 17. 已存储的安全和非安全寄存器 . . . 16
图 18. SFR 视图显示带有安全和非安全地址空间的外设 . . . 17
图 19. 新建空项目 . . . 17
图 20. 目标选择 . . . 18
图 21. 项目设置 - 选择空项目 . . . 19
图 22. 空项目结构 . . . 20
图 23. 项目设置 - 选择 STM32Cube . . . 20
图 24. 生成的项目来自 STM32CubeMX . . . 21